

2012

IBM System z Technical University

Enabling the infrastructure for smarter computing

CICS Transaction Server *zFS Usage and Best Practices*

zAI25

Steve Zemblowski



Notes

This presentation will cover CICS use of the Unix Systems Services File Systems (HFS and ZFS). The installation and resource usage requirements will be discussed. Best practices for the use of zFS and CICS will be explored.

Copyright and Trademarks

© IBM Corporation 2012. All Rights Reserved.

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide.

Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at “Copyright and trademark information” at

www.ibm.com/legal/copytrade.shtml.

Notes

Copyrights and trademarks.

Session Agenda

- z/OS Unix File System
- CICS usage of zFS
 - ✓ Installation
 - ✓ Execution
- CICS Bundles
 - ✓ Usage
 - ✓ Management
- Best practices
- Summary

Notes

The following topics will be presented during this session:

This section will do a high level review of the z/OS Unix File System looking at the directory structures that CICS will require.

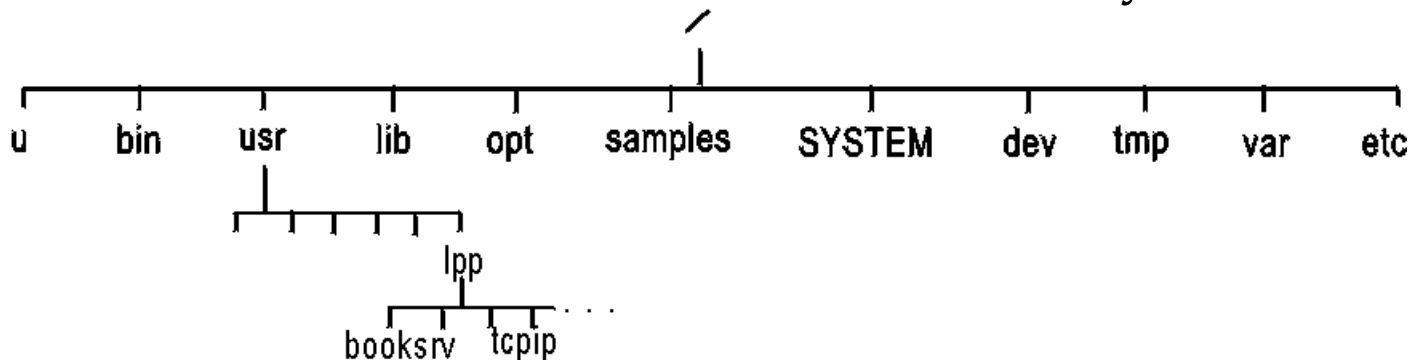
The use of various directories and files by CICS will be examined. Directories required and used by the installation process as well as during the execution of CICS will be explained.

The CICS BUNDLE resource will be reviewed as well as techniques to manage the bundle artifact across various execution environments (Test, Production) will be discussed.

Best practices for CICS and zFS will also be discussed.

What is the z/OS Unix File System?

- Hierarchical file system
 - ✓ Files
 - Programs or data
 - ✓ Directories
 - Files
 - Other directories
 - ✓ Additional file systems local or remote
 - Mounted on directories of the root file system



Notes

The z/OS UNIX file system, like other UNIX systems, is a hierarchical file system consisting of the root file system and all the file systems that are added to it. Files are members of a directory, and each directory is in turn a member of another directory at a higher level. The highest level of the hierarchy is the root directory. Each instance of the system contains only one root directory.

An hierarchical file system consists of the following:

Files, which contain data or programs. A file containing a load module or shell script or REXX program is called an executable file. Files are kept in directories.

Directories that contain files, other directories, or both. Directories are arranged hierarchically, in a structure that resembles an upside-down tree, with the root directory at the top and the branches at the bottom. The root is the first directory for the file system at the top of the tree and is designated by a slash (/).

Additional local or remote file systems, which are mounted on directories of the root file system or of additional file systems.

What is the z/OS Unix File System...

- High level directory structure used by CICS
 - ✓ /usr/lpp
 - Directory for z/OS elements or features
 - ✓ /var
 - Directory for dynamic data created by z/OS elements
 - ✓ /u/CICS *region userid*
 - Working directory for a CICS region

Notes

This chart shows the high level directory structure required by CICS during installation and execution

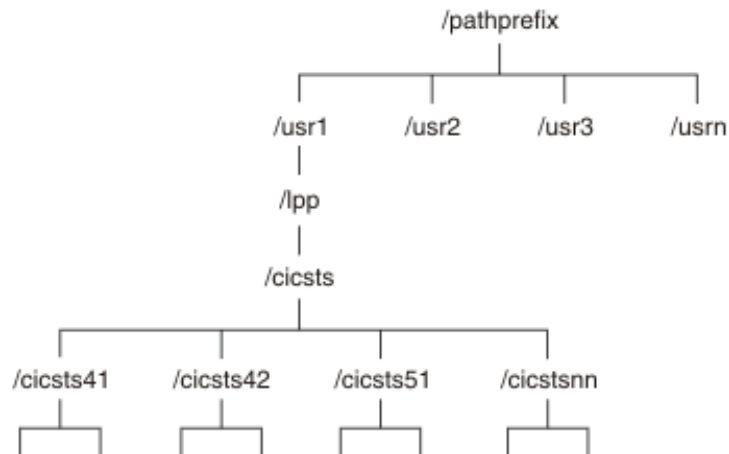
Under the `/usr/lpp` directory are directories for z/OS elements and features.

`/var`, which contains dynamic data that is used internally by products and by elements and features of z/OS. Any files or directories that are needed are created when code is executed or customized. An example of this is caching data.

Each CICS region will have a working directory under the `/u/CICS region userid` structure.

CICS Installation Directories

- Create release specific directory structure
 - ✓ /usr/lpp/cicsts is created ONCE for all releases
 - ✓ Depending on SMP/E target zone structure
 - You may create variants of /cicsts51



Notes

CICS TS for z/OS, Version 5.1 requires PDSE data set support for installation to complete successfully. DFSMS/MVS, which is supplied as an element of z/OS, provides this support for PDSE data sets. The components concerned are part of the CICS support for Java applications.

For FMID JCI680D, which contains the z/OS UNIX-dependent code, the OMVS address space must be active in full-function mode, otherwise the installation of this function fails.

In the set of installation jobs, some initial jobs create the z/OS UNIX files and the directories shown. Run these jobs before any of the normal DFHINSTn jobs. The user ID running these jobs requires superuser authority.

If you normally maintain additional SMP/E target zone libraries to apply service, you can also create additional directories at the /cicsts51 level to create the z/OS UNIX equivalent

CICS Installation Directories...

- Directories under /cicsts51
 - ✓ docs
 - ✓ IBM
 - ✓ JVMPROFILES
 - ✓ lib
 - ✓ pipelines
 - ✓ samples
 - ✓ schemas
 - ✓ ...

Notes

This chart shows some of the sub directories under the /cicsts51 directory.

CICS Execution zFS Usage

- System Initialization
 - ✓ USSHOME
 - The name and path of the root directory for CICS files
 - Default: /usr/lpp/cicsts/cicsts51
 - ✓ JVMPROFILEDIR
 - The directory name for the JVM profiles
 - Default: /usr/lpp/cicsts/cicsts51
 - Profiles DFHJVMAX and DFHOSGI must always be available to CICS

Notes

The **USSHOME** system initialization parameter specifies the name and path of the root directory for CICS Transaction Server files on z/OS UNIX.

The **JVMPROFILEDIR** system initialization parameter specifies the name (up to 240 characters long) of a z/OS UNIX directory that contains the JVM profiles for CICS. CICS searches this directory for the profiles it needs to configure JVMs.

The default value of **JVMPROFILEDIR** is the same as the default value of the **USSHOME** system initialization parameter, which specifies the name and path of the root directory for CICS files on z/OS UNIX, followed by the subdirectory **JVMProfiles**. Changing the **USSHOME** parameter has no effect on the **JVMPROFILEDIR** parameter value.

If you want CICS to load the JVM profiles from a different directory, you must complete one of the following tasks:

Change the value of the **JVMPROFILEDIR** system initialization parameter.
Link to your JVM profiles from the directory specified by **JVMPROFILEDIR** using UNIX soft links. Use this method to store your JVM profiles in any place in the z/OS UNIX file system.

The supplied sample JVM profiles **DFHJVMAX** and **DFHOSGI** must always be available to CICS in the directory that is specified by **JVMPROFILEDIR**, or linked to using UNIX soft links from that directory.

CICS Execution zFS Usage...

- DOCTEMPLATE resource definition
 - ✓ HFSFILE attribute
 - Fully qualified (absolute) name
 - Relative to CICS region home directory (USSHOME)
 - CICS region must have read permissions (directory and file)
- URIMAP resource definition
 - ✓ HFSFILE attribute
 - Fully qualified (absolute name)
 - Relative to CICS region home directory (USSHOME)
 - CICS region must have read permissions (directory and file)

Notes

DOCTEMPATE

HFSFILE(hfsfile)

When the template resides in a z/OS® UNIX System Services file, this specifies the fully qualified (absolute) or relative name of the z/OS UNIX file. The name can be specified as an absolute name including all directories and beginning with a slash,

URIMAP

HFSFILE(name)

This attribute is for USAGE(SERVER), where a static response is to be provided.

HFSFILE specifies the fully qualified (absolute) or relative name of a z/OS UNIX System Services HFS file that forms the body of the static response that is sent to the HTTP request from the web client. You can specify the name as an absolute path including all directories and beginning with a slash.

CICS Execution zFS Usage...

- WEBSERVICE resource definition
 - ✓ Defines the attributes of a web service request
 - ✓ WSBIND
 - zFS file name of the web service binding file
 - ✓ WSDLFILE
 - zFS file name of the WSDL file
 - ✓ ARCHIVEFILE
 - zFS file name of the zip file that contains the WSDL

Notes

A **WEBSERVICE** resource defines aspects of the run time environment for a CICS application program deployed in a Web services setting, where the mapping between application data structure and SOAP messages has been generated using the CICS Web services assistant. Although CICS provides the usual resource definition mechanisms for **WEBSERVICE** resources, they are typically installed dynamically, using the output produced by the assistant.

WSBIND(hfsfile)

Specifies the 1-255 character fully-qualified file name of the Web service binding file on z/OS UNIX.

WSDLFILE(hfsfile)

Specifies the 1-255 character fully-qualified file name of the Web service description (WSDL) file on z/OS UNIX. This file is used when full runtime validation is active.

ARCHIVEFILE(hfsfile)

Specifies the 1-255 character fully-qualified file name of an archive that contains one or more WSDL files. The supported format for the archive is .zip.

CICS Execution zFS Usage...

- Web Services PIPELINE
 - ✓ Defines quality of service for web service requests
 - ✓ CONFIGFILE
 - zFS file name that defines processing nodes
 - Samples provided
 - ✓ SHELF (/var/cicsts/)
 - Directory for CICS to store installed artifacts
 - May be shared by multiple CICS regions
 - ✓ WSDIR
 - Directory for WSDL and WSBIND files that are to be installed into CICS
 - Unique for each PIPELINE in a CICS region

Notes

A PIPELINE resource is used when a CICS application is in the role of a Web service provider or requester. It provides information about the message handler programs that act on a service request and on the response.

CONFIGFILE(name)

Specifies the name of a z/OS UNIX file that contains information about the processing nodes that will act on a service request, and on the response.

SHELF(/var/cicsts/| directory)

Specifies the 1-255 character fully-qualified name of a directory (a shelf, primarily for Web service binding files) on z/OS UNIX.

CICS regions into which the PIPELINE definition is installed must have full permissions to the shelf directory – read, write, and the ability to create subdirectories.

A single shelf directory can be shared by multiple CICS regions and by multiple PIPELINE definitions. Within a shelf directory, each CICS region uses a separate subdirectory to keep its files separate from those of other CICS regions. Within each region's directory, each PIPELINE uses a separate subdirectory.

After a CICS region performs a cold or initial start, it deletes its subdirectories from the shelf before trying to use the shelf.

You should not attempt to modify the contents of a shelf that is referred to by an installed PIPELINE definition. If you do, the effects are unpredictable.

WSDIR(directory)

specifies the 1-255 character fully-qualified name of the Web service binding directory (also known as the pickup directory) on z/OS UNIX. Each PIPELINE installed in a CICS region must specify a different Web service binding directory.

CICS Execution zFS Usage...

- ATOMSERVICE resource definition
 - ✓ Describes the attributes of an Atom Service Request
 - Type of document returned, source of data, formatting
 - ✓ CONFIGFILE
 - zFS file name that describes metadata for the feed request
 - ✓ BINDFILE
 - zFS file name that describes the XML transformation for a FEED or COLLECTION request

Notes

You use the Atom configuration wizard in CICS Explorer to create an Atom configuration file. This file is comprised of a number of XML elements, which provide metadata for the Atom feed. You enter the basic details to create the file using the New wizard and then add additional information using the editor.

CONFIGFILE(name)

Specifies the fully qualified (absolute) or relative name of an Atom configuration file stored in z/OS UNIX System Services. The Atom configuration file contains XML that specifies metadata and field names for the Atom document that is returned for this resource definition.

The name can be specified as an absolute path including all directories and beginning with a slash, for example, /u/atom/myfeed.xml. Alternatively, it can be specified as a path relative to the HOME directory of the CICS region user ID, for example, atom/myfeed.xml (with no leading forward slash). Up to 255 characters can be used.

BINDFILE(name)

Specifies the fully qualified (absolute) or relative name of an XML binding stored in z/OS® UNIX System Services. This attribute is not used for an Atom service or category document. You create an XML binding using the CICS XML assistant program DFHLS2SC.

For resource types FILE and TSQUEUE, the XML binding is required, and it specifies the data structures used by the resource named in RESOURCENAME, which supplies the data for the Atom document.

For resource type PROGRAM, an XML binding is optional, and you create it using the resource that the program accesses to obtain the data for the Atom entries, not the program itself. You must specify an XML binding for resource type PROGRAM if you are using the resource handling parameters in the DFHATOMPARGS container to pass information from the Atom configuration file to the program. If you are not doing this, do not specify an XML binding.

The name of the XML binding can be specified as an absolute path including all directories and beginning with a slash, for example, /u/atom/atomicstest.xsdbind. Alternatively, it can be specified as a path relative to the HOME directory of the CICS® region user ID; for example, atom/atomicstest.xsdbind (with no leading forward slash). Up to 255 characters can be used.

CICS Execution zFS Usage...

- Java resources
 - ✓ PROGRAM definition
 - JVMCLASS
 - Name of the main class in a Java program
 - ✓ JVMSERVER definitions
 - JVMPROFILE
 - zFS file name in JVMPROFILE directory

Notes

JVMCLASS(class)

Specifies the name of the main class in a Java program.

For OSGi bundles that run in a JVM server, this value is the name of the OSGi service. The OSGi service is registered when you install the BUNDLE resource that contains the OSGi bundle. You can look up the name of the OSGi service in the Bundle Parts view in CICS Explorer.

For Java programs that run in a JVM server, this value is the class name qualified by the package name.

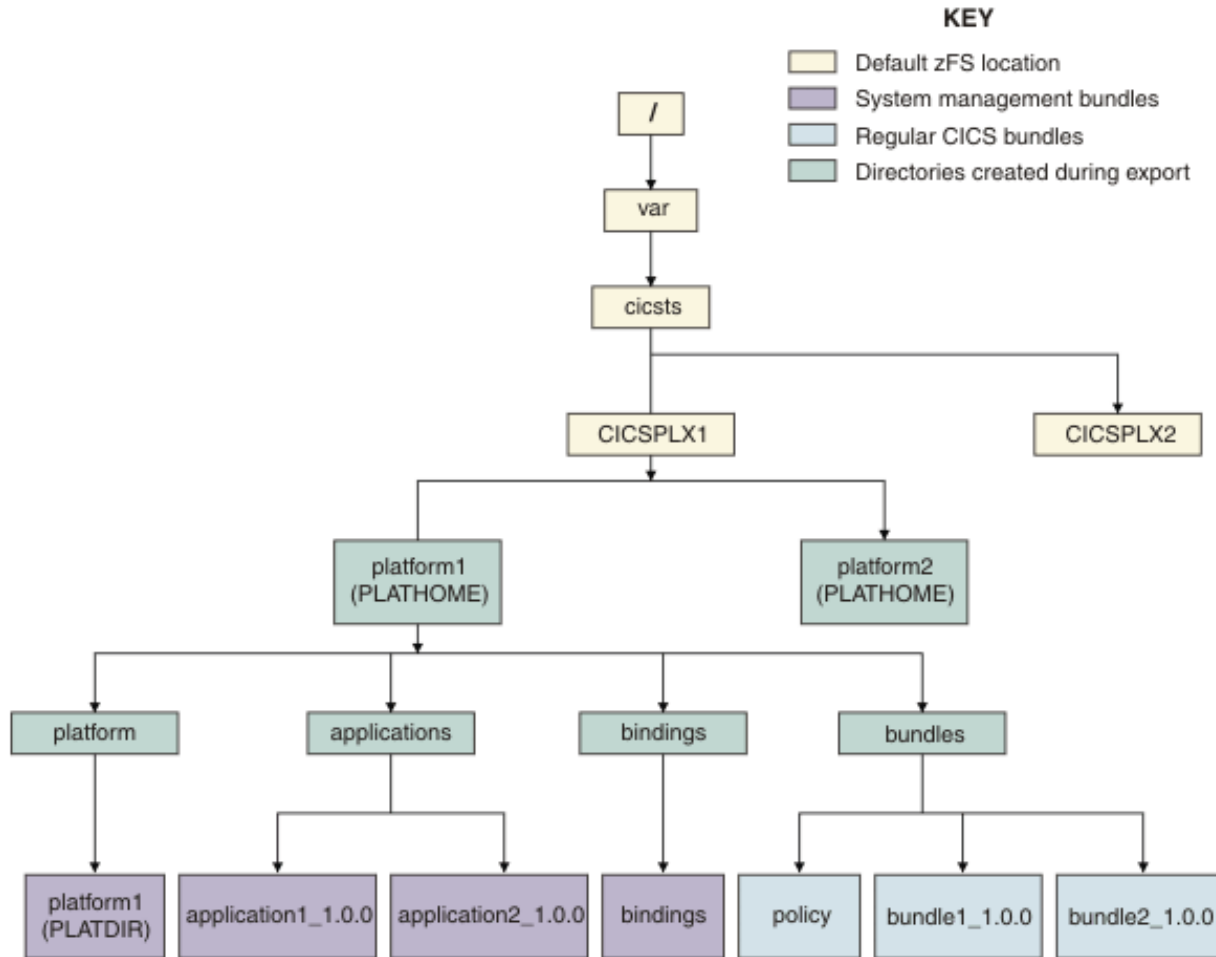
The names are case sensitive and must be entered with the correct combination of uppercase and lowercase letters. If you use a terminal, ensure that uppercase translation is suppressed.

JVMPROFILE(jvmprofile)

Specifies the 1 - 8 character name of the JVM profile for the JVM server. The JVM profile is a file in the z/OS UNIX directory that is specified by the system initialization parameter JVMPROFILEDIR. Alternatively, the file can be in another place in the z/OS UNIX file system and be referenced by a UNIX soft link from the JVMPROFILEDIR directory. The profile contains the JVM options for running a JVM server.

CICS Execution zFS Usage...

- Platform directory structure



Notes

You can package a CICS platform into a platform bundle to deploy, manage, and monitor it as a single entity. With a platform bundle, you can install and manage the resources for the platform in all of the CICS regions in a platform.

A CICS platform bundle describes the content of a platform that is created without the need for a connection to CICS. A platform definition describes the location of a platform bundle and is persisted in the CICSplex SM data repository. A platform definition is a CICSplex SM definition that is used to install a platform bundle from zFS. In CICS, we provide a Platform as a Service (PaaS) which means the delivery of a computing platform, including applications, optimized middleware, development tools, and Java™ and Web 2.0 runtime environments, as a service over the Internet.

This management bundle format is a different format from a standard CICS bundle and is common across application, platform, and binding bundles. Creation and export of CICS platform bundles is managed through the CICS Explorer in a similar way to the creation and export of a standard CICS bundle.

CICS BUNDLE Resource

- Collection of CICS resources
 - ✓ Managed as a logical unit
 - All bundle parts are managed together
 - If any part is disabled, all parts of the bundle are disabled
 - Bundle artifact is migrated from test to production
 - Similar to a load module
 - ✓ Resource dependencies can be specified
 - e.g. FILE(ABC) must be present for the bundle to install

Notes

A **BUNDLE** resource defines a bundle, a unit of deployment for an application. A bundle is a collection of CICS resources, artifacts, references, and a manifest that you can deploy into a CICS region to represent an application.

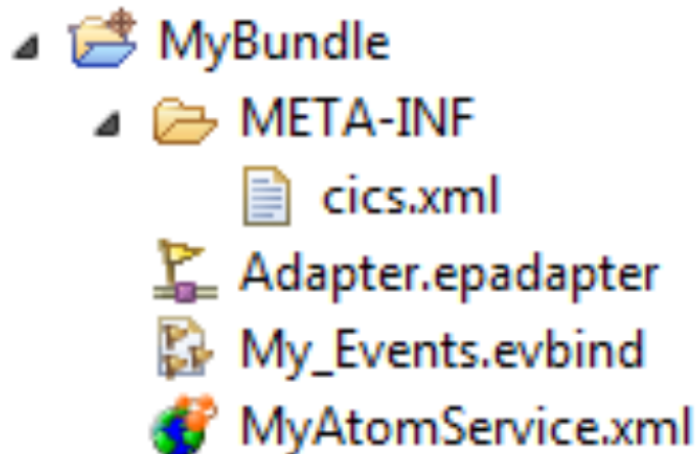
The manifest is a file that describes the contents of the bundle, including what resources to create in the CICS region and the location of the supporting artifacts, what prerequisites are required for the application to run successfully, and any services that the application can offer to other applications.

If CICS fails to create one or more of the application resources, the **BUNDLE** installs in a **DISABLED** state. You can use the IBM® CICS Explorer to view the state of each resource. You can try to enable the **BUNDLE** resource again. However, if one of the resources, for example a **WEBSERVICE**, installs in an **UNUSABLE** state, you cannot enable the **BUNDLE** resource. You must discard the **BUNDLE** resource and re-create the definition.

If you disable one of the resources that was created by the **BUNDLE**, for example an **EVENTBINDING** resource, CICS disables the **BUNDLE** resource as well. However, any other resources that are part of the bundle remain in an enabled state in the CICS region. If you reenable the resource successfully, the **BUNDLE** resource also changes to the **ENABLED** state. If you try to discard a disabled **BUNDLE** resource when enabled resources that belong to the bundle are in the CICS region, CICS issues a message and the discard fails. You must disable each of the enabled resources before discarding the **BUNDLE** resource. You can use the **DISABLE BUNDLE** command on a disabled bundle to disable all of the associated resources.

CICS BUNDLE Resource...

- BUNDLE resource definition
 - ✓ BUNDLEDIR points to zFS Bundle artifact
- Bundle artifact
 - ✓ Created by the CICS Explorer
 - XML file deployed to zFS

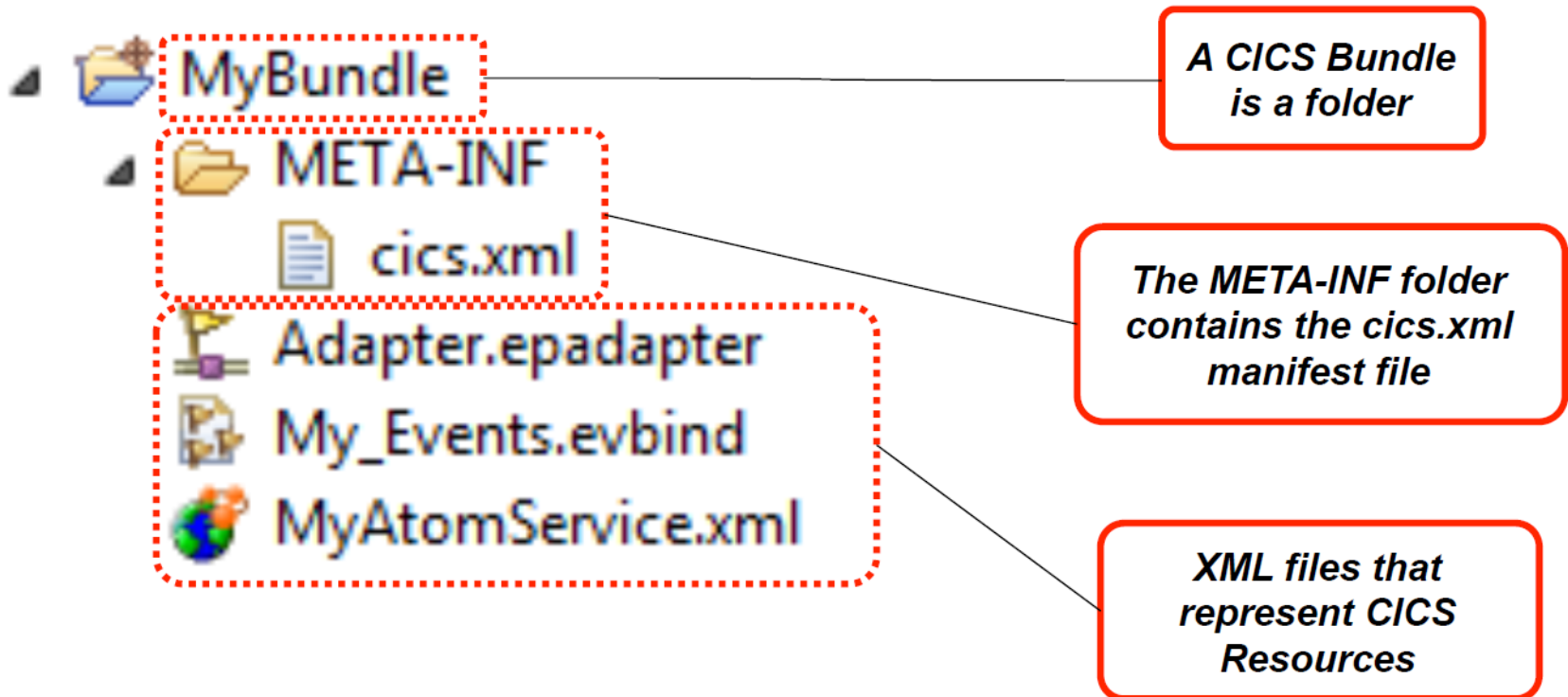


Notes

A CICS bundle is a directory that contains artifacts and a manifest that describes the bundle and its dependencies. The bundle is the unit of deployment for an application. CICS bundles provide a way of grouping and managing related resources. CICS bundles also provide versioning for the management of resource updates, and can declare dependencies on other resources outside the bundle. Application developers can use CICS bundles for application packaging and deployment, business events, and services. System programmers can use CICS bundles for system events and policies.

CICS BUNDLE Resource...

- What's in a CICS Bundle?

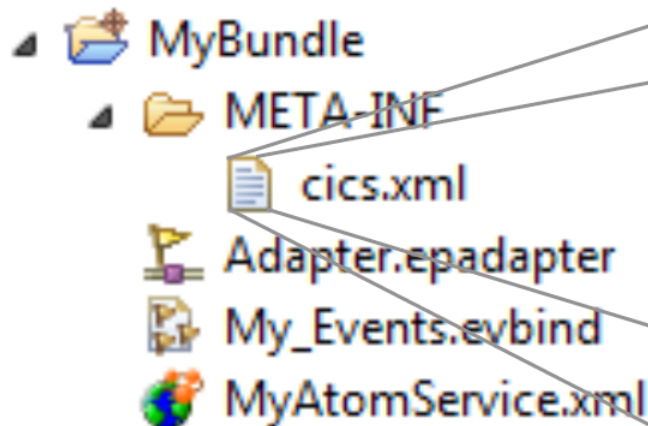


Notes

The contents of the CICS Bundle artifact.

CICS BUNDLE Resource...

- What's in the cics.xml manifest file?



```
<define name="Adapter" type="http://  
www.ibm.com/xmlns/prod/cics/bundle/  
EPADAPTER" path="Adapter.epadapter"/>  
  
<define name="My_Events"  
type="http://www.ibm.com/xmlns/prod/  
cics/bundle/EVENTBINDING"  
path="My_Events.evbind"/>  
  
<define name="MyAtomService"  
type="http://www.ibm.com/xmlns/prod/  
cics/bundle/ATOMSERVICE"  
path="MyAtomService.xml"/>  
  
<import name="MYFILE" type="http://  
www.ibm.com/xmlns/prod/cics/bundle/  
FILE" optional="false" warn="true"/>
```

Notes

The contents of the `cics.xml` manifest file.

CICS BUNDLE Resource...

- What's in the rest of the bundle file?

Definitions for all the resources in the CICS Bundle

```

<define name="Adapter"
type="http://www.ibm.com/xmlns/
prod/cics/bundle/EPADAPTER"

<define name="My_Events"
type="http://www.ibm.com/xmlns/
prod/cics/bundle/EVENTBINDING"
path="My_Events.evbind"/>

<define name="MyAtomService"
type="http://www.ibm.com/xmlns/
prod/cics/bundle/ATOMSERVICE"
path="MyAtomService.xml"/>

```

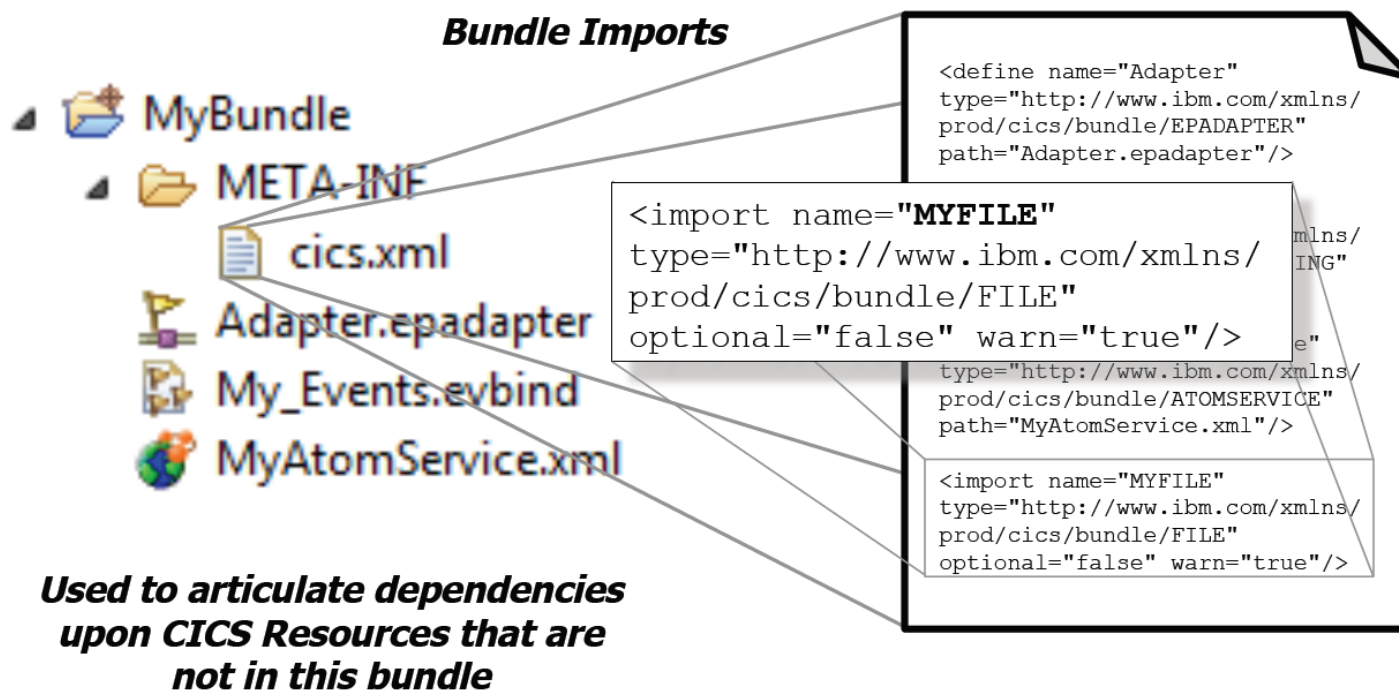
The CICS Bundle will not install if any of the defined bundle parts are missing from the bundle folder

Notes

The DEFINE contents of the bundle file.

CICS BUNDLE Resource...

- How are resource dependencies specified?

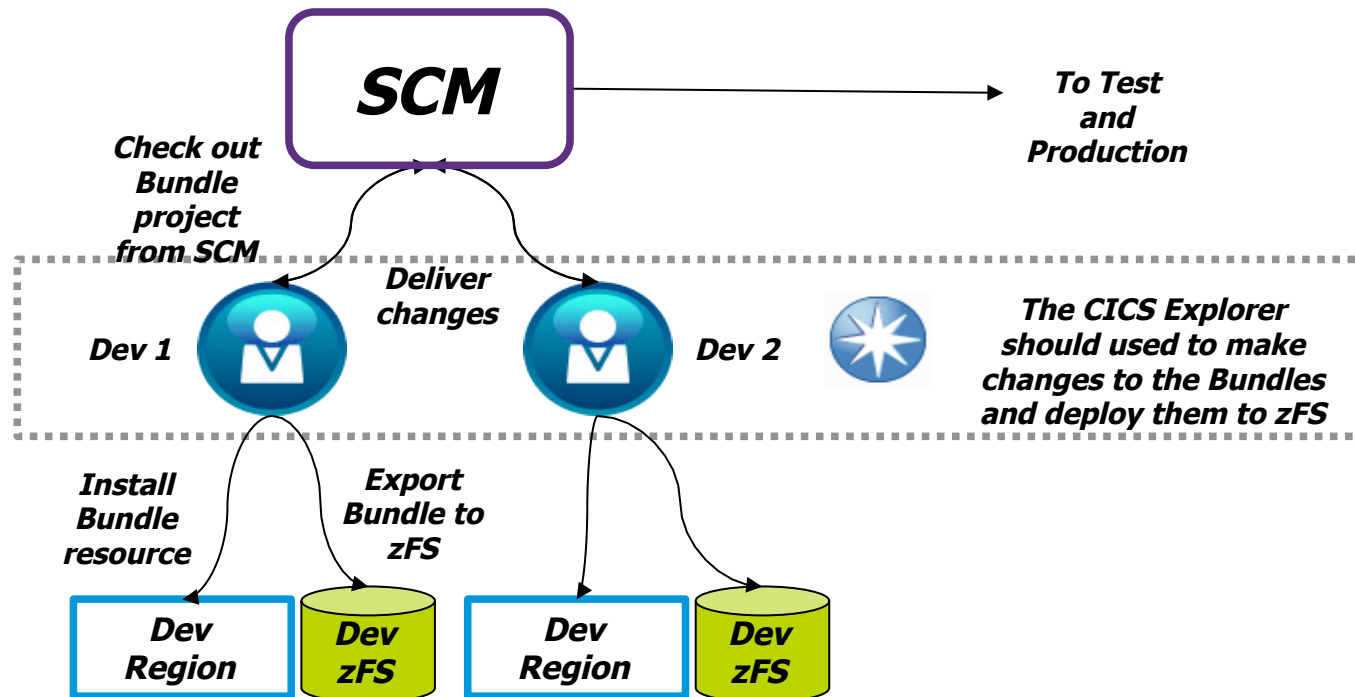


Notes

The contents of the bundle showing a resource dependency.

CICS Bundle Management

- Managing changes to CICS Bundles
 - ✓ CICS Bundle project should be treated as source code
 - ✓ Changes should be managed and shared using a source code management (SCM) repository

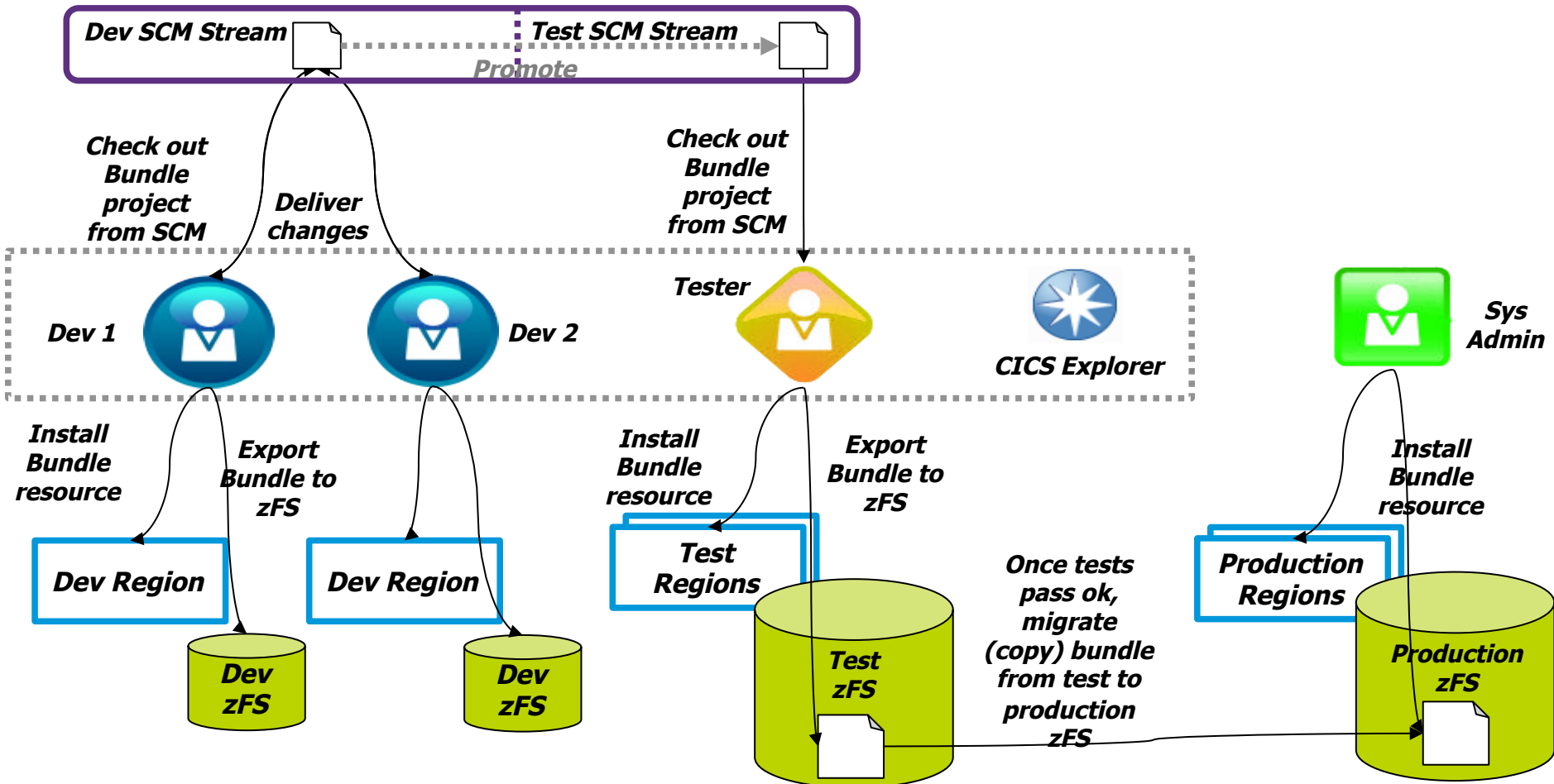


CICS Bundle Management...

- Migrating CICS Bundles from Dev to Test to Production
 - ✓ BUNDLES should be treated like any other CICS resource that has a reference to an artifact that lies outside the CSD
 - PROGRAMS have load modules/java classes
 - WEBSERVICES have wsbind files
 - ✓ You should migrate the CICS Bundle on zFS before the BUNDLE resource
 - You wouldn't migrate a new PROGRAM resource before you migrated the load module for it!

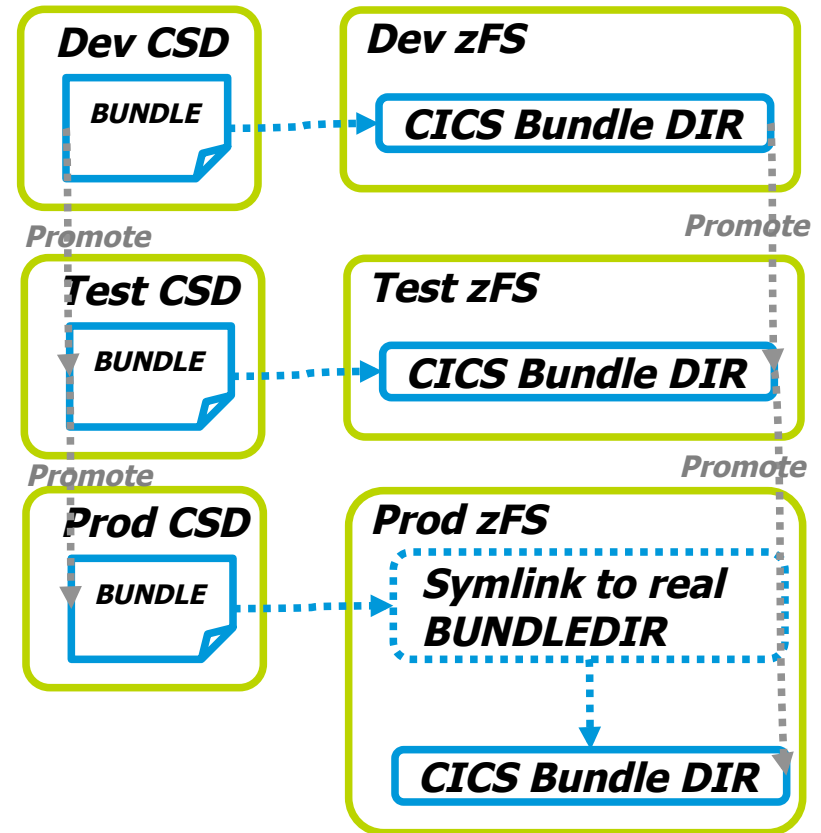
CICS Bundle Management...

- Migrating CICS Bundles from Dev to Test to Production



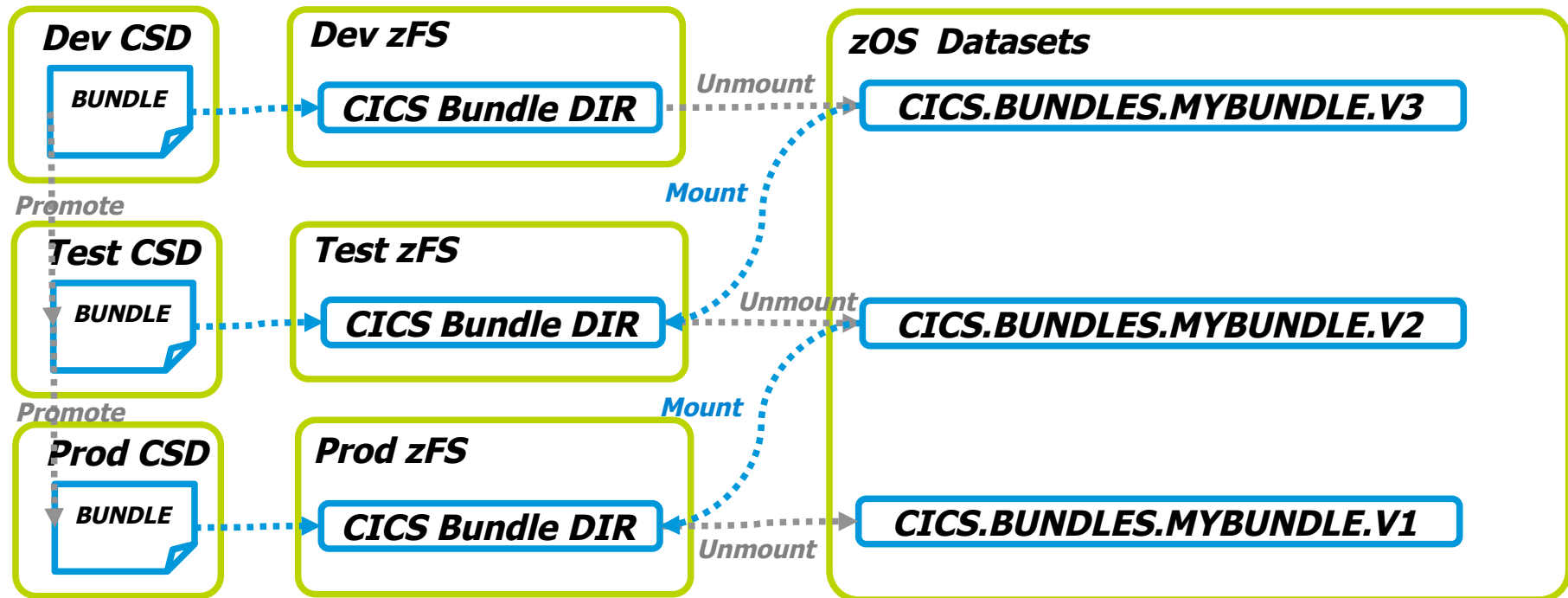
CICS Bundle Management...

- Avoid having to change BUNDLEDIR
 - ✓ Changing the BUNDLE resource's BUNDLEDIR is undesirable because you aren't promoting the same resource that you tested
 - ✓ Option 1: Put your CICS Bundle XML in the same directories on each zFS
 - ✓ Option 2: Use Symlinks to point to the real bundle location



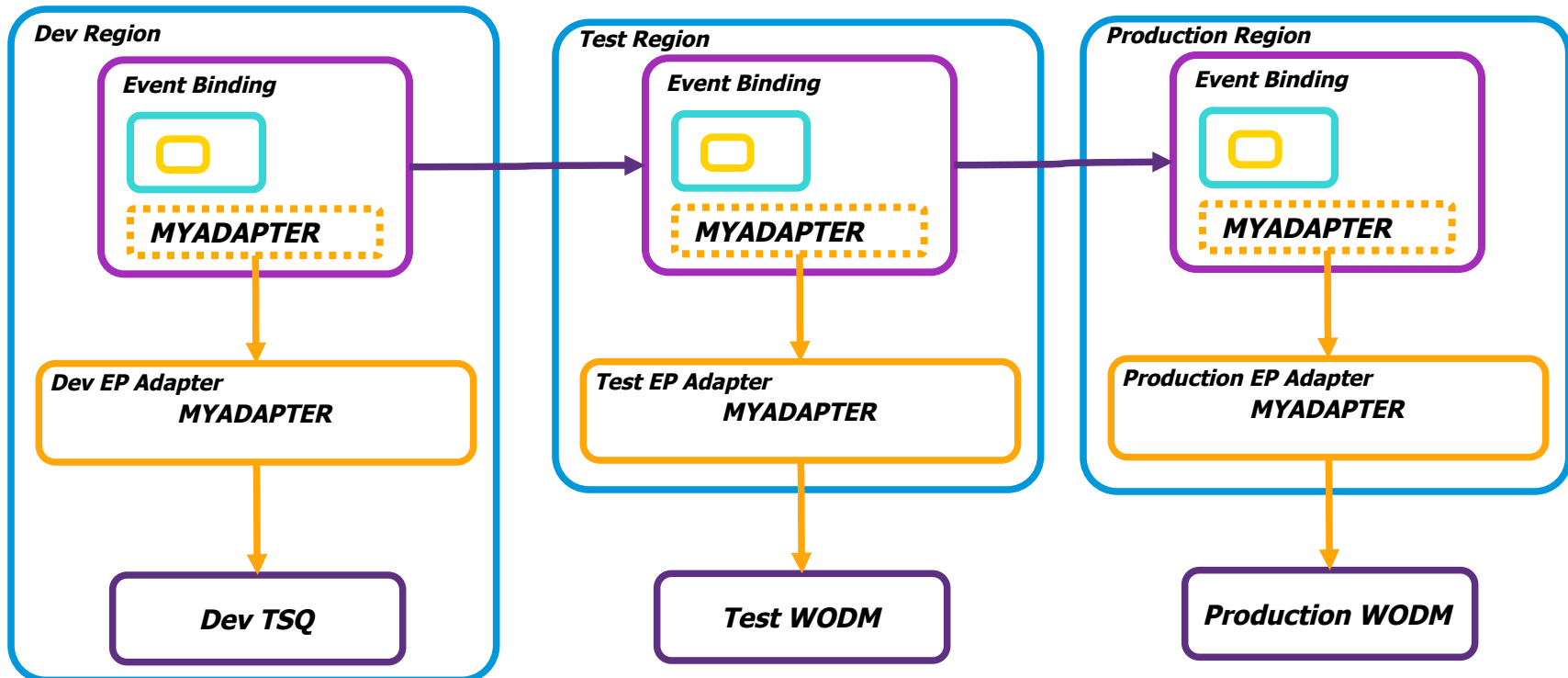
CICS Bundle Management...

- Avoid having to change BUNDLEDIR
 - Option 3: Use zFS mounts to migrate bundles



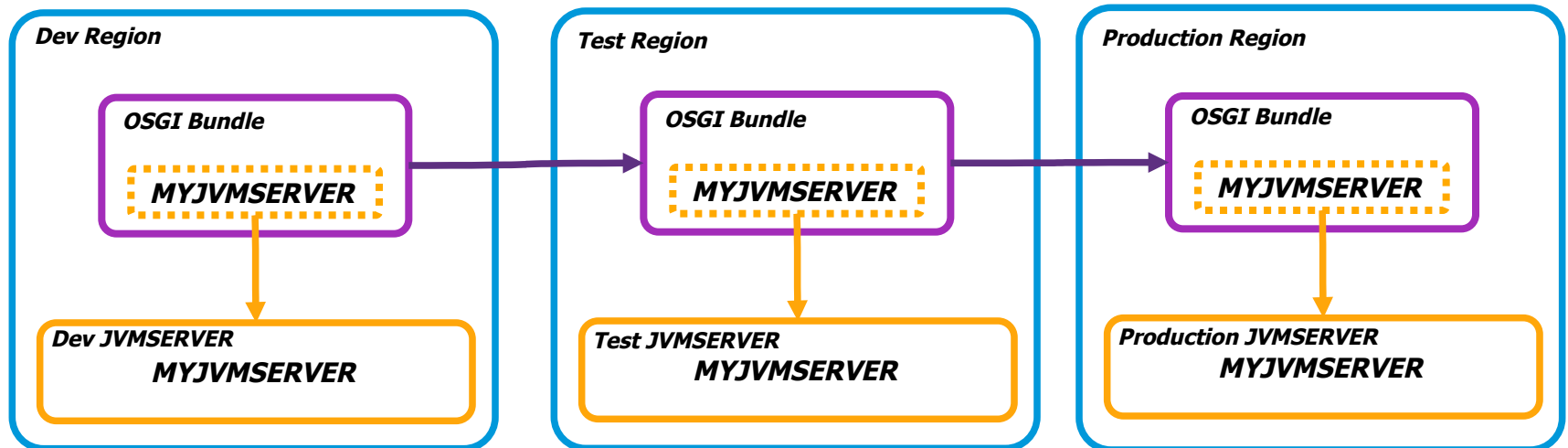
CICS Bundle Management...

- Use separate EPADAPTERs on all EVENTBINDINGS



CICS Bundle Management...

- Use the same JVMSERVER names in all regions
 - For CICS TS V4.2 have the same JVMSERVERs in all regions to ensure no changes are needed to CICS Bundles containing OSGI Bundles during migration



Notes

The preceding charts describe CICS best practices for bundles.

Best Practice: zFS Setup

1. Create data set for usage as /var/cicsts zFS
- 2a If using shared zFS across a sysplex
 - Mount data set onto root filing system as /cicsts as a r/w filing system
 - On each LPAR create symbolic link to link /var/cicsts to /cicsts (/var is always a symlink to /<LPAR>/var)
 - > ln -s /cicsts /var/cicsts
- 2b. If using non-shared zFS,
 - Mount data set onto /var as /var/cicsts

Best Practice: zFS Setup...

3. Set permissions of /var/cicsts to allow access by multiple readers (CICS regions) and a common writer (administrator)

1. Set the owner to have read/write/execute, this will be the userid required by zFS to export files into zFS

2. Set the readers to have read/execute access

```
> chgrp -R <group> /cicsts
```

```
> chmod -R 750 /cicsts
```

Best Practice: zFS Setup

4. Set default file permission for the FTP daemon to give writers(owners) rw and readers(group) r

- i.e UMASK 027

5. If write access is required by multiple groups of writers then you can either

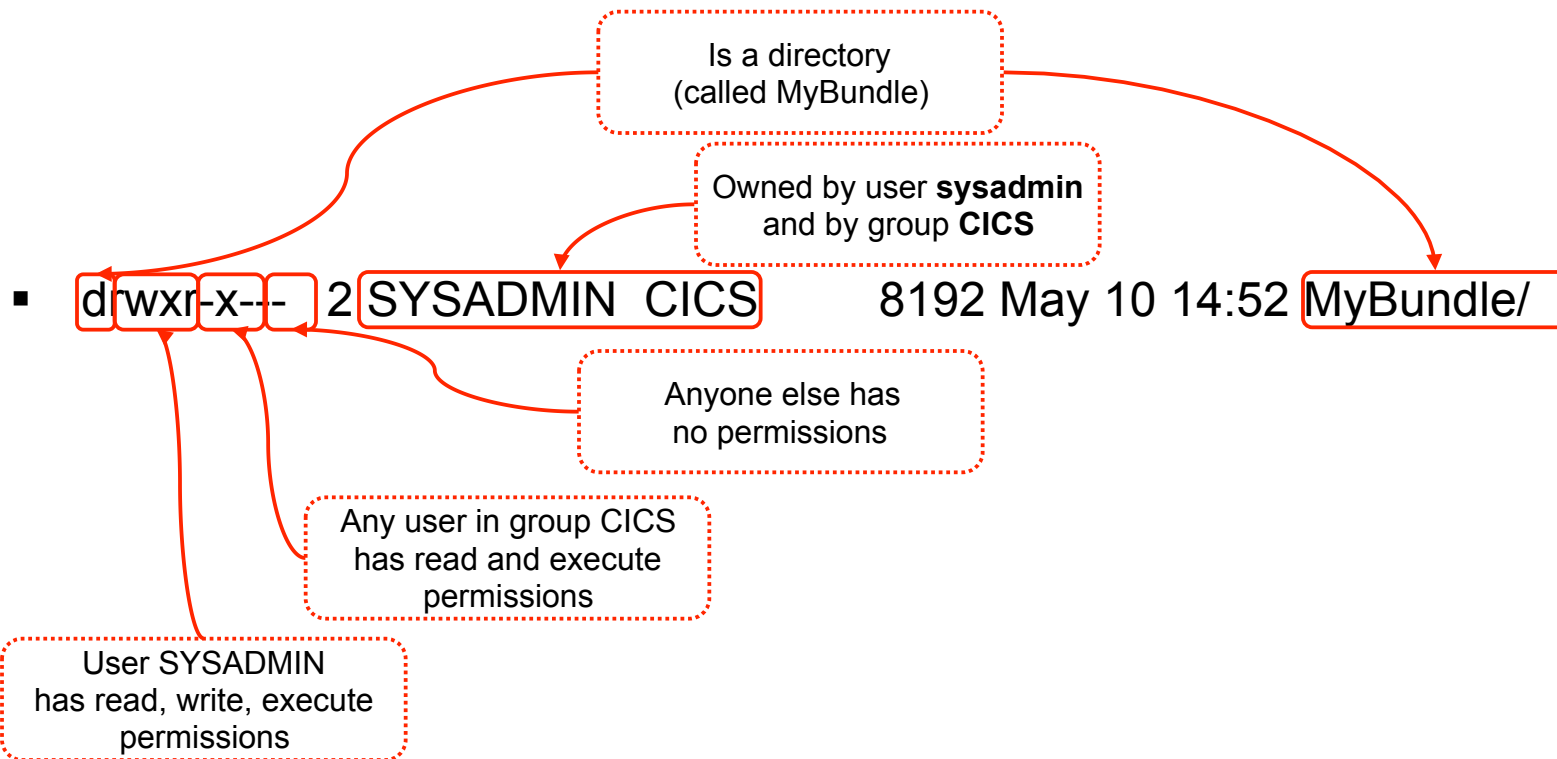
a) Set the group ownership to a common group in which all the writers are members, this will then limit you to running all the readers (CICS regions) under the same uid

Set the FTP UMASK to 207, to give write permission to the group

b) Use ACLs to add additional group permissions. This can be achieved by activating the FSSEC resource class and using the setfacl command

Best Practice: zFS Security

- Use the UNIX permission flags for Owner, Group and All to control access to your CICS Bundle XML on zFS



Best Practice: zFS Security...

- Typically you would have the following permissions for a Bundle XML

- CICS Bundle Directory
 - RWX - Owner (Sys Admin/Person who exported the CICS Bundle XML)
 - R-X - Group (The group that all your CICS regions belong to)
 - --- - All

- CICS Bundle XML Files
 - RWX - Owner (Sys Admin/Person who exported the CICS Bundle XML)
 - R-- - Group (The group that all your CICS regions belong to)
 - --- - All

Best Practice: zFS Security...

- A user can be in many groups, but a file has only one group permission
 - Meaning that if multiple users need to access the file they must be in that group, and will all share the same permissions
 - This means 2 logical groups of users (such as system admins and CICS regions) can not use UNIX permission bits to be granted access

- ACLs provide a solution to this as they allow a more flexible model
 - Multiple groups to a have permissions
 - ACL inheritance to be controlled
 - However, they may only restrict the access permissions that are defined by the UNIX permissions bits

Best Practice: Performance

- z/OS V1R11 provides local read caching – removing overheads for reads
- z/OS V1R13 provides direct I/O for read and write, removing need to function ship these commands to the LPAR owning the file system

Summary

- What does CICS need from zFS for installation?
- What does CICS read and write to zFS during execution?
- What is a CICS Bundle and how is it used?
- How do I manage CICS Bundles and other artifacts?
- What are some of the CICS zFS best practices?

Google us or check us out at:

 ibm.com/developerworks/cicsdev

 facebook.com/IBMCICS


 twitter.com/IBM_CICS


 youtube.com/cicsfluff

 youtube.com/cicsexplorer

 themasterterminal.com

 twitter.com/IBM_System_z

 [CICS Explorer Forum ibm.com/developerworks/forums/forum.jspa?forumID=1475&start=0](http://ibm.com/developerworks/forums/forum.jspa?forumID=1475&start=0)

 [CICS-L list Forum
listserv.uga.edu/archives/cics-l.html](http://listserv.uga.edu/archives/cics-l.html)



I ♥ CICS